Benchmarking von Workflow-Management-Systemen

(Kurzbeitrag)

Michael Gillmann, Peter Muth, Gerhard Weikum, Jeanine Weissenfels

Universität des Saarlandes, Fachbereich Informatik
Postfach 151150, D–66041 Saarbrücken
E-mail: {gillmann, muth, weikum, weissenfels}@cs.uni-sb.de
WWW: http://www-dbs.cs.uni-sb.de

Zusammenfassung

Heute verfügbare Workflow-Management-Systeme (WfMS) weisen in ihrer Leistungsfähigkeit und Skalierbarkeit deutliche Defizite auf. Während an Verbesserungen intensiv gearbeitet wird, sollten Leistungsvergleiche zwischen verschiedenen WfMS generell auf eine systematische, objektive Basis gestellt werden. Zu diesem Zweck entwickelt dieser Kurzbeitrag einen Benchmark für WfMS, der – ausgehend von den Basisaktivitäten des TPC-C-Benchmarks – in verschiedenen Ausbaustufen Stärken und Schwächen von WfMS ausloten und systematisch vergleichen kann. Die Benchmark-Definition wägt sorgfältig ab zwischen Einfachheit, Realitätsnähe und Aussagekraft von Benchmark-Messungen und ist in dem im Industriestandard UML verwendeten State- und Activity-Chart-Formalismus spezifiziert.

1 Einleitung

Die Workflow-Technologie steht auf dem Sprung, zu einem wichtigen Pfeiler der informationstechnischen Infrastruktur moderner Unternehmen zu werden [4, 8, 11, 16, 17, 28, 29]. Ein Workflow-Management-System, kurz: WfMS, ist in der Lage, verschiedenste, zu einem Geschäftsprozeß gehörende Aktivitäten zu koordinieren, indem es Aktivitäten gemäß einer Kontroll- und Datenflußspezifikation für den vorliegenden Prozeßtyp startet. Aktivitäten können ihrerseits externe Applikationen aufrufen, ggf. über Dienste der zugrundeliegenden Middleware (z.B. Corba). Das Spektrum der so unterstützten Prozesse reicht von sogenannten Produktions-Workflows mit ausschließlich vollautomatischen Aktivitäten bis zu komplexen, kooperativen Workflows mit einem signifikanten Anteil an intellektuell-interaktiv zu bearbeitenden Aktivitäten.

In der Workflow-Technologie steckt einerseits ein sehr großes Potential, andererseits aber weisen die kommerziell verfügbaren WfMS immer noch wesentliche Mängel hinsichtlich ihrer Administrierbarkeit, Interoperabilität und vor allem ihrer Leistung auf [1, 2]. Eine ausgesprochene Schwachstelle ist insbesondere die fehlende Skalierbarkeit hinsichtlich des Durchsatzes gleichzeitig aktiver Workflows und der gleichzeitigen Bedienung vieler Benutzer. An der Überwindung dieser Defizite wird in Forschung und Entwicklung intensiv gearbeitet (siehe z.B. [5, 10, 13, 19, 22, 25, 31]). Idealerweise sollte der Erfolg dieser Bemühungen objektiv meßbar sein, so daß künftige leistungskritische Anwendungsprojekte bereits im Vorfeld eine zielgerichtete Auswahl des von seiner Performance am besten geeigneten WfMS treffen können. An diesem Punkt setzt der vorliegende Kurzbeitrag ein, indem er einen systematisch konstruierten Benchmark vorschlägt, mit dessen Hilfe die Stärken und Schwächen verschiedener WfMS aufgedeckt und die Leistungsfähigkeit der Systeme weitgehend objektiv verglichen werden kann.

Diese Arbeit wurde im Rahmen des Projektes “Architektur, Konfiguration und Administration großer Workflow–Management–Systeme” der Deutschen Forschungsgesellschaft (DFG) erstellt.

Das Benchmarking von Computersystemen generell und Informationssystemen im besonderen wird weithin als eine Art schwarzer Kunst angesehen [9, 15, 21]. Bereits die Definition eines Benchmarks ist alles andere als einfach. Es gilt dabei, sich einerseits so nah wie möglich an realistischen Anwendungen zu orientieren und andererseits die Lastspezifikation des Benchmarks so einfach wie möglich zu halten. Einfachheit soll sich dabei in zweierlei Hinsicht auszahlen: der Benchmark sollte möglichst einfach auf einer breiten Palette angebotener Produkte implementierbar bzw. schnell portierbar sein, und die Resultate der Benchmark-Messungen sollten einfach interpretierbar sein und zu klaren Vergleichsaussagen über verschiedene Systeme führen. Die langjährige Arbeit, insbesondere des Transaction Processing Performance Council (TPC [26]), auf dem Gebiet der Datenbank- und Transaktions-Benchmarks zeigt, daß die Definition eines Benchmarks zahlreiche subtile Fallen beinhalten kann und die Entwicklung eines systematischen, hinreichend praxisnahen und zugleich einfachen Benchmarks eine echte Herausforderung ist [9].

Für den hier betrachteten Bereich der WfMS steckt das Benchmarking in den Kinderschuhen. Die einzigen uns bekannten Arbeiten, die sich mit dem Thema intensiver befaßt haben, sind sehr spezieller Natur: in [18] wird ein einziges System (Lotus Notes) betrachtet, das eher als Dokumentenverwaltungs- und -kommunikationssystem denn als WfMS zu bezeichnen ist, und es werden nur Leistungsaspekte der E-Mail-Komponente untersucht; in [3] steht die spezielle Anwendung von DNS-Laboruntersuchungen für die Genomforschung im Blickpunkt, wobei ein funktional spezialisierter WfMS-Prototyp zugrundeliegt, und der vorgeschlagene Benchmark eben auch nur extrem spezialisierte Leistungsaspekte abdeckt; in [6] wird mit Hilfe des TPC-D Benchmarks ein komplettes Anwendungssystem (SAP R/3) statt eines isolierten Datenbanksystems untersucht; in [7] schließlich wird ein Benchmark für aktive Datenbanksysteme vorgeschlagen, der aber keine spezifischen Elemente hinsichtlich Workflow-Anwendungen enthält. Der vorliegende Kurzbeitrag ist somit mehr oder weniger die erste Arbeit, die versucht, einen allgemeinen, systematischen Benchmark für WfMS zu definieren. Wir orientieren uns dabei an einem ausgefeilten Electronic-Commerce-Szenario, dessen Grundgerüst – Bestellung, Lieferung, Zahlung – aus dem bekannten TPC-C-Benchmark abgeleitet wird. Gegenüber TPC-C ist jedoch der Kontroll- und Datenfluß zwischen den Aktivitäten berücksichtigt, und es werden weitergehende Aktivitäten eingeführt, die die Funktionalität und Leistungsfähigkeit eines WfMS ausloten sollen. Hierbei betrachten wir jedoch ausschließlich die Ressourcen, die unmittelbar vom WfMS verwaltet werden (z.B. für Verwaltungsdaten), aber nicht die, die von den Aktivitäten benötigt werden. Die wissenschaftliche Schwierigkeit bei der Entwicklung des Benchmarks lag im sorgfältigen Abwägen zwischen Einfachheit, Realitäts- und Praxisnähe und der Systematik in Aufbau und Aussagekraft des Benchmarks. Der Benchmark ist sowohl abstrakt, d.h. sprachunabhängig, als auch vollständig als State- und Activity-Chart spezifiziert, also einer Teilsprache des Industriestandards UML. Eine Portierung in die verschiedenen Spezifikationssprachen kommerzieller WfMS ist relativ einfach möglich. Messungen mit konkreten WfMS-Produkten und -Prototypen sind in Vorbereitung.

Der Rest dieses Kurzbeitrags ist wie folgt aufgebaut. Abschnitt 2 identifiziert die zur Leistungsbewertung eines WfMS relevanten Metriken. Abschnitt 3 diskutiert Anforderungen an einen systematischen WfMS-Benchmark und führt verschiedene Bausteine ein, aus denen der Benchmark konstruiert ist. Abschnitt 4 definiert den vorgeschlagenen Benchmark, zunächst in Gestalt eines informalen Szenarios, dann als formale Spezifikation mit State- und Activity-Charts. Abschnitt 5 schließlich diskutiert die Praxistauglichkeit des Benchmarks unter verschiedenen Aspekten.

2 Bewertungsmetriken

Aus betriebswirtschaftlicher Sicht steht beim Einsatz von WfMS die Steigerung der Effizienz von Geschäftsprozessen im Vordergrund. Dies umfaßt jedoch nicht nur monetäre oder zeitliche Größen. Wichtig sind auch Aspekte wie Kundenzufriedenheit, Motivation und Engagement der Mitarbeiter, gleichmäßige Mitarbeiterauslastung ohne wesentliche Überlastung usw.. Eine quantitative Erfassung dieser Aspekte ist jedoch schwierig. Dagegen zielen *betriebswirtschaftliche Kennzahlsysteme* auf größere Transparenz rein betrieblicher Aspekte ab [24]. Die Verwendung von WfMS erlaubt die

direkte Erfassung entsprechender Kenngrößen, die bisher nicht explizit verfügbar waren. Beispielsweise können die Mitarbeiterauslastung und die mittlere Bearbeitungszeit ganzer Geschäftsprozesse mit geringem Aufwand erfaßt und statistisch aufbereitet werden.

Um den Einfluß der Performance verschiedener WfMS auf die Effizienz von Geschäftsprozessen systematisch und reproduzierbar zu bestimmen, müssen *Leistungsmaße für WfMS* definiert werden. Einerseits sollen diese Leistungsmaße sehr eng an den obigen Qualitätskriterien für Geschäftsprozesse angelehnt sein, um leicht interpretierbare Werte zu erhalten, andererseits sollen für unseren Zweck WfMS-externe Parameter wie zum Beispiel die Auswahl und Leistungsfähigkeit der Sachbearbeiter keine Rolle spielen. Ferner müssen die Leistungsmaße systemunabhängig festgelegt sein, damit sie für jedes untersuchte WfMS definiert sind und dieselbe Bedeutung haben.

Da ein einzelnes Leistungsmaß nicht alle Anforderungen gleichzeitig erfüllen kann, schlagen wir eine Hierarchie von Leistungsmaßen vor, die von globalen, "makroskopischen" Metriken bis zu Detailgrößen reicht:

(1) Durchsatz ganzer Workflows sowie die Durchlaufzeit vollautomatisierter Produktions-Workflows
(2) Durchsatz und Antwortzeit einzelner Workflow-Schritte, z.B. das Starten intellektuell-interaktiv zu bearbeitender Aktivitäten
(3) Durchsatz und Antwortzeit systeminterner Primitive, z.B. Anzahl der Ereignisse oder Nachrichten, die pro Zeiteinheit abgearbeitet werden können
(4) Ressourcenverbrauch, z.B. Plattenzugriffe, Hauptspeichernutzung, Netzbelastung, etc.

Das Messen globaler Leistungsmetriken wie Durchsatz und Durchlaufzeit ganzer Workflows erfordert die Bereitstellung einer vollständigen Systemumgebung zum Ausführen des Workflows und die genaue Festlegung einer Reihe von Parametern. Interaktive Aktivitäten erfordern Benutzerinteraktion und können daher nicht unmittelbar Teil eines Benchmarks sein. Sie müssen simuliert werden. In diesem Fall ist eine Verteilungsfunktion zur stochastischen Definition der Dauer der interaktiven Aktivitäten festzulegen.

3 Bausteine eines WfMS-Benchmarks

In diesem Abschnitt diskutieren wir, aus welchen Bausteinen sich ein WfMS-Benchmark zusammensetzen sollte und welche Eigenschaften eines WfMS damit analysiert werden können.

Kontrollflußkonstrukte. Die Hauptaufgabe eines WfMS besteht in der Koordinierung des Kontrollflußes zwischen einzelnen Aktivitäten. Der Kontrollfluß ist typischerweise nicht linear, sondern enthält Konstrukte wie Verzweigungen, Schleifen und Parallelität. Um ihre Auswirkungen auf das Systemverhalten bestimmen zu können, sollte ein Benchmark das volle Spektrum der Kontrollflußkonstrukte enthalten.

Interaktive und vollautomatische Aktivitäten. Eine weitere Aufgabe von WfMS ist die Integration von Anwendungen, die die einzelnen Aktivitäten eines Workflows implementieren. Wir unterscheiden zwischen interaktiven und vollautomatischen, d.h. ohne Benutzerinteraktion ablaufenden Aktivitäten, da beide Arten in der Regel unterschiedlich vom WfMS gehandhabt werden und ein unterschiedliches Laufzeitverhalten aufweisen. Darüber hinaus sind viele WfMS auf einen Workflow-Typ (vollautomatisch oder interaktiv) optimiert.

Während vollautomatische Aktivitäten häufig kurze Laufzeiten aufweisen, wird die Laufzeit der interaktiven Aktivitäten hauptsächlich durch den menschlichen Bearbeiter bestimmt. Durch die "Denkzeiten" der Bearbeiter belastet eine interaktive Aktivität relativ zu ihrer Gesamtdauer das WfMS weniger als eine vollautomatische Aktivität. So werden Effekte des WfMS auf das Durchlaufzeitverhalten der Workflows weniger relevant. Andererseits können unter Umständen Ressourcen für Verwaltungsdaten lange belegt werden, so daß sie für die Verwaltung anderer Aktivitäten nicht mehr zur Verfügung stehen und der Systemdurchsatz vermindert wird. Die Auswirkungen solcher Aspekte sollen in die Ergebnisse des Benchmarks eingehen, d.h. sowohl vollautomatische als auch interaktive Aktivitäten bilden weitere Bausteine unseres Benchmarks.

Schnittstelle zu externen Anwendungen. Um auch die Schnittstellen zu externen Anwendungen in die Last miteinbeziehen zu können, sind die Aufrufe von Anwendungen durch das WfMS und der Austausch von (kontrollflußrelevanten) Daten weitere Bestandteile des Benchmarks.

Fazit. Aus den bisherigen Überlegungen leiten wir ab, daß der Benchmark zwei unterschiedliche Workflow-Typen beinhalten soll:

(1) Einen vollautomatischen Produktions-Workflow ohne menschliche Interaktion. Durch die kurze Dauer der vollautomatischen Aktivitäten wird die Last, die im System durch Kontextwechsel, Zustandsübergänge, Kommunikation, etc. verursacht wird, dominant. Relevante globale Leistungsmaße sind sowohl der Durchsatz als auch die Durchlaufzeit ganzer Workflows

(2) Einen komplexen und teilweise interaktiven Workflow. Dieser Typ dient zur Hervorhebung der Ressourcenverwaltung bei lange laufenden Workflows, z.B. während einer langen interaktiven Aktivität. Das relevante globale Leistungsmaß ist der Workflow-Durchsatz. Darüber hinaus ist bei der Detailanalyse die Antwortzeit einzelner Interaktionsschritte (z.B. das Starten einer Aktivität durch den Benutzer) interessant.

4 Definition des WfMS-Benchmarks

Der in diesem Abschnitt vorgeschlagene Benchmark enthält alle in Abschnitt 3 beschriebenen Bausteine und ermöglicht die Ermittlung der in Abschnitt 2 genannten Leistungsmaße. Der Benchmark basiert auf einem realistischen Szenario aus dem Bereich des Electronic Commerce.

4.1 Szenario

Bei der Spezifikation unseres Benchmarks orientieren wir uns, ähnlich dem TPC-C Benchmark für Transaktionssysteme, an der Gesamtverarbeitung einer Bestellung wie sie in einem großen Versandhaus vorkommt. Der Unterschied zum TPC-C Benchmark ist jedoch, daß die Systemlast nicht durch die einzelnen Aktivitäten, wie Bestellungen, Lieferungen etc. bestimmt wird, sondern Kontroll– und Datenfluß zwischen den Aktivitäten vom WfMS gehandhabt werden müssen, und so die Last erzeugt wird. Ferner wird der Aufruf einzelner Aktivitäten des Workflows explizit berücksichtigt.

Kontrollfluß. Der Benchmark beginnt mit einer Aktivität, die eine *neue Bestellung* in einer Datenbank erfaßt. Das Versandhaus stellt zwei Arten von Zahlungsmodalitäten zur Verfügung. Der Kunde kann entweder bei der Bestellung seine Kreditkartennummer angeben oder eine Rechnung anfordern, nach deren Erhalt er die angefallene Rechnungssumme überweisen muß. Der Kontrollfluß muß an dieser Stelle aufgespaltet werden. Wurde eine *Kreditkartenzahlung* gewünscht, wird zunächst eine Aktivität zur Bonitätsprüfung der Kreditkarte gestartet. Nach der Prüfung der Kreditkarte werden die beiden Kontrollfluß-Pfade wieder zusammengeführt.

Nach der Registrierung der Bestellung werden der *Versand* sowie das Senden einer *Bestätigungs-Mail* an den Kunden initiiert. Dazu werden parallel zwei Sub-Workflows gestartet. Einer der beiden Sub-Workflows startet eine Aktivität, die eine Bestätigung an den Kunden versendet. Das Versenden von Waren erfolgt extern aus einem oder mehreren Lagern, die u. U. sogar von separaten Großhändlern autonom verwaltet werden. Daher startet der zweite Sub-Workflow zunächst eine Aktivität, die für jedes Lager die auszuliefernden Artikel bestimmt. Die Zuordnung wird in Artikellisten festgehalten. Innerhalb eines jeden Lagers, aus dem Artikel für die aktuelle Bestellung ausgeliefert werden müssen, wird eine Anwendung gestartet, die als Parameter die Artikelliste erhält und eine Versendegarantie zurückliefert. In einer Schleife werden so oft einzelne Lager angesprochen, bis alle Artikel auf der Bestellung bearbeitet wurden und die Versendung garantiert wurde. Das WfMS muß das Ende der Schleife registrieren, bei Bezahlung per Rechnung einen Zähler für eventuelle Mahnungen initialisieren und sich den Stichtag für das mögliche Versenden der ersten Mahnung merken. Wenn beide Subworkflows beendet sind, wird der Kontrollfluß erneut aufgespalten, abhängig davon, welche Zahlungsmodalität gewünscht wurde. Wurde Zahlung per Kreditkarte gewählt, wird die Kreditkarte nun belastet und der Workflow beendet. Auf diesem Kontrollfluß-Pfad werden keinerlei menschliche Eingriffe benötigt.

Soll eine Rechnung gestellt werden, wird eine Aktivität gestartet, die auf den *Eingang der Zahlung* wartet. Diese Aktivität gilt als intellektuell-interaktiv, da die Eingabe von Daten, z.B. ob der Rechnungsbetrag vollständig eingegangen ist, durch einen Mitarbeiter des Versandhauses erfolgt. Ist die Zahlung erfolgt, ist das Szenario beendet. Sollte die Rechnung nicht innerhalb einer Frist von 14 Tagen beglichen sein, wird dem Kunden eine Mahnung geschickt. Dieser Vorgang wiederholt sich maximal dreimal im Abstand von 14 Tagen, bis die Rechnung beglichen ist. Wurde die Zahlung 14 Tage nach der dritten Mahnung noch immer nicht geleistet, so wird die Rechtsabteilung des Versandhauses informiert, deren Vorgehen aber nicht mehr Teil unseres Benchmarks ist. Das Mahnverfahren wird in einer Schleife spezifiziert. Innerhalb der Schleife wird eine Aktivität gestartet, die eine *Mahnung* versendet, der Stichtag für das Versenden der nächsten Mahnung ermittelt und der Zähler für die bereits versendeten Mahnungen inkrementiert.

Datenfluß. Alle Eingabedaten werden in der Aktivität gesammelt, die die Bestellung erfaßt, und von ihr (bedarfsweise) an die übrigen Aktivitäten verteilt. Es ergibt sich ein sternförmiger Datenfluß, ausgehend von dieser Aktivität. Lediglich innerhalb der Schleife, die die Artikel auf die Lager verteilt, gibt es zusätzlichen Datenfluß, nämlich die jeweiligen Artikellisten in die eine und die Versandgarantien in die andere Richtung.

4.2 Exkurs über State- und Activity-Charts

Der State- und Activity-Chart-Formalismus wurde in den Arbeiten von David Harel [12, 14] entwikkelt und inzwischen in den Industrie-Standard UML [27] übernommen. State- und Activity-Charts stellen duale Sichtweisen einer Spezifikation dar. Ein *Activity-Chart* spezifiziert den Datenfluß zwischen den Aktivitäten in Form eines gerichteten Graphen mit Aktivitäten als Knoten und Datenelementen als Kantenbeschriftung.

State-Charts beschreiben das Verhalten einer Spezifikation, indem sie den Kontrollfluß zwischen Aktivitäten als Transitionen zwischen Zuständen spezifizieren. Ein State-Chart ist im wesentlichen ein endlicher Automat mit einem ausgewiesenen Anfangszustand und Transitionen, die mit Event-Condition-Action Regeln (ECA-Regeln) beschriftet sind. Eine Transition von Zustand *X* nach Zustand *Y*, die mit der ECA-Regel *E[C]/A* beschriftet ist, feuert genau dann, wenn sich der Automat in Zustand *X* befindet, das Ereignis *E* eintritt und die Bedingung C erfüllt ist. Der Automat wechselt dann in den Zustand *Y* und die Aktion *A* wird ausgeführt. Bedingungen und Aktionen werden durch Variablennamen identifiziert, wobei auch diejenigen, die zur Spezifikation des Datenflußes im entsprechenden Activity-Chart benutzt wurden, verwendet werden können. Darüber hinaus kann eine Aktion sowohl explizit eine Aktivität starten (durch den Ausdruck *st!(aktivität))*, als auch Ereignisse erzeugen oder Bedingungen setzen (z.B. *fs!(C)* setzt die Bedingung *C* auf den Wert falsch). Jeder der drei Teile des *E[C]/A*-Tripels kann auch leer sein.

Zwei weitere Besonderheiten unterscheiden State-Charts von herkömmlichen Zustandsautomaten. Zum einen ist dies die im Hinblick auf schrittweise Verfeinerung und Abstraktion wichtige Möglichkeit zur Schachtelung von Zuständen. Ein Zustand kann ein komplettes State-Chart beinhalten, wobei beim Betreten eines übergeordneten Zustands implizit der jeweilige initiale Unterzustand betreten wird und beim Verlassen eines Elternzustandes alle betretenen Kinderzustände verlassen werden. Zum anderen erlauben State-Charts orthogonale Zustandskomponenten, mit denen Parallelität ausgedrückt wird, so daß Transitionen in verschiedenen orthogonalen Komponenten gleichzeitig feuern können.

4.3 Benchmark-Spezifikation als State-Chart (UML)

Im folgenden präsentieren wir das State-Chart und das Activity-Chart der vollständigen Spezifikation unseres Benchmarks. Der Übersichtlichkeit halber haben wir generell auf die Darstellung von Ausnahmebehandlungen verzichtet (z.B. Nichtlieferbarkeit bestellter Artikel). Abbildung 1 zeigt das State-Chart. Initial wird die Aktivität *BestellungErfassen* gestartet und der Workflow geht in den Zustand *NeueBestellung_S* über. Die Bedingung *KreditkartenZahlung* spezifiziert die gewünschte Zahlungsmodalität und wird von der Aktivität *BestellungErfassen* gesetzt. Ist die Bedingung wahr,

wird die Aktivität *KreditkartePrüfen* gestartet, die bei ihrer Terminierung die Bedingung *KreditkarteOk* auf wahr oder falsch setzt. Schlägt der Test fehl, wird der Workflow beendet, ansonsten geht er in den geschachtelten Zustand *Versand_S* über. Innerhalb dieses geschachtelten Zustandes werden zwei Sub-Workflows ausgeführt. Die Parallelität der Ausführung wird durch die gestrichelte Linie spezifiziert. In den Sub-Workflows werden die Aktivitäten *Bestätigen*, *LagerErmitteln* und *LagerInstruieren* gestartet. Sie setzen jeweils die Bedingungen *BestätigungGesendet*, *LagerErmittelt* und *VersandGarantiert* auf wahr. Dabei durchlaufen die Sub-Workflows die Zustände *Bestätigen_S*, *LagerErmitteln_S* und *LagerInstruieren_S* und terminieren in den Zuständen *BestätigungGesendet_S* beziehungsweise *Abgeschickt_S*. Vor Betreten des Zustandes *Abgeschickt_S* werden der Zähler für verschickte Mahnungen *#Mahnungen* und der *Stichtag* des Versendens der ersten Mahnung initialisiert.

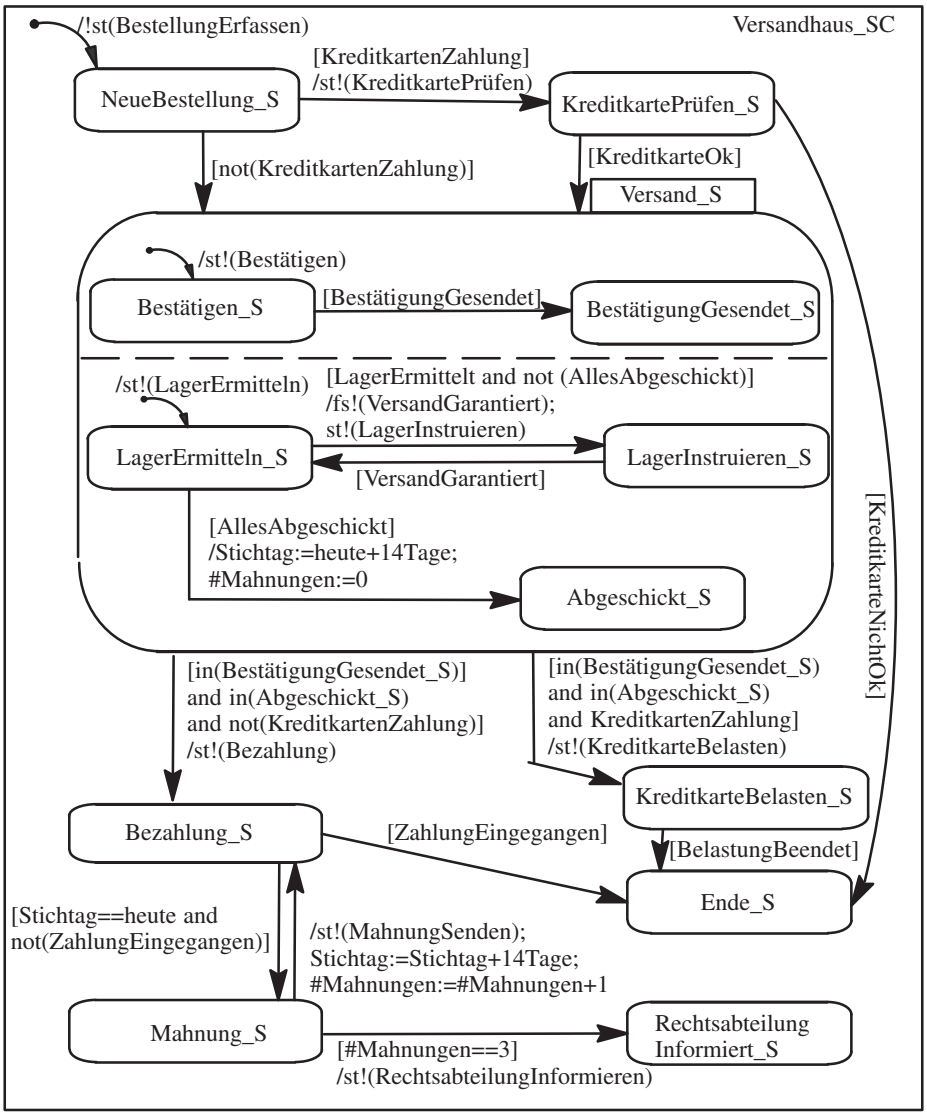

Abbildung 1: State-Chart des WfMS-Benchmarks

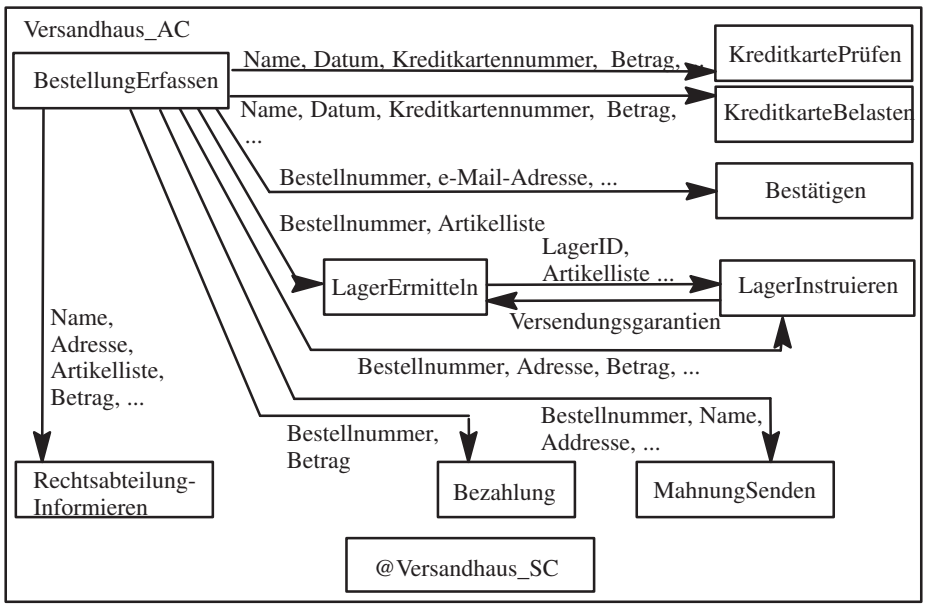

Abbildung 2: Activity-Chart des WfMS-Benchmarks

Haben beide Sub-Workflows ihren Endzustand erreicht, angezeigt durch die Bedingungen *in(Abgeschickt_S)* und *in(BestätigungGesendet_S)*, wird der geschachtelte Zustand *Versand_S* verlassen. Abhängig vom Wert der Bedingung *KreditkartenZahlung* wird die Kreditkarte durch Starten der Aktivität *KreditkarteBelasten* belastet und der Workflow endet im Zustand *Ende_S,* oder die Aktivität *Bezahlung* wird gestartet und der Workflow geht in den Zustand *Bezahlung_S* über. Bei der Aktivität *Bezahlung* wird die Bedingung *ZahlungEingegangen* auf wahr gesetzt, sobald die Rechnung beglichen wurde. Dadurch geht der Workflow in den Zustand *Ende_S* über. Die Aktivitäten *MahnungSenden* und *RechtsabteilungInformieren* mit den korrespondierenden Zuständen *Mahnung_S* und *RechtsabteilungInformiert_S* implementieren das Mahnverfahren.

Abbildung 2 skizziert das zugehörige Activity-Chart. Das Activity-Chart enthält einen Zeiger auf das korrespondierende State-Chart (*@Versandhaus_SC*) sowie alle an dem Workflow beteiligten Aktivitäten. Alle Eingabedaten werden in der Aktivität *BestellungErfassen* gesammelt und von ihr bedarfsweise an die übrigen Aktivitäten verteilt. Zusätzlich übermittelt die Aktivität *LagerErmitteln* der Aktivität *LagerInstruieren* eine *LagerID* mit der zugehörigen Artikelliste. Die Aktivität *LagerInstruieren* schickt an ihrem Ende der Aktivität *LagerErmitteln* die erhaltenen Versandgarantien zurück.

4.4 Benchmark-Parameter

Ein komplexer Benchmark ist von einer Reihe von Parametern abhängig. Für unseren Benchmark sind dies die Ankunftsrate neuer Workflows, deren Eingabeparameter sowie die Dauer simulierter Aktivitäten.

Wir nehmen an, daß neue Workflows mit exponential verteilter Zwischenankunftszeit eintreffen. Der Erwartungswert der Zwischenankunftszeit bestimmt die Systemlast und kann für Skalierbarkeitstests variiert werden. Die Eingabeparameter wie z.B. die Zahlungsart bestimmen den Ablauf des Workflows. Hier bietet sich eine diskrete Bernoulli-Verteilung zwischen Bezahlung per Kreditkarte und per Rechnung an. Die Zahl der in einer Bestellung involvierten Lager sowie die Zahl der notwendigen Mahnungen bestimmt die Zahl der entsprechenden Schleifendurchläufe im Workflow. Wir nehmen hier jeweils (linksseitig abgeschnittene) Normalverteilungen an. Workflow-Aktivitäten

können real ausgeführt oder simuliert werden. Für jede simulierte Aktivität muß eine Verteilungsfunktion zur Festlegung der Dauer ihrer Ausführung bestimmt werden (z.B. Normalverteilung).

5 Praxistauglichkeit des Benchmarks

5.1 Aussagekraft

Die in Abschnitt 3 für einen Benchmark als wichtig identifizierten Kontrollflußkonstrukte Verzweigungen, Schleifen und Parallelität sind im vorgeschlagenen Benchmark enthalten. Der Benchmark besteht aus einem vollautomatisch ohne Benutzerinteraktion zu durchlaufenden Zweig, sowie aus einem Zweig mit interaktiven Aktivitäten. Durch die Wahl des Anteils der Kreditkartenzahlungen läßt sich die Leistungsfähigkeit des zu untersuchenden Systems daher sowohl für automatisch ablaufende Workflows als auch für interaktive Workflows bestimmen. Auch eine gemischte Last ist möglich. Durch Variation der Ankunftsrate neuer Bestellungen kann die Skalierbarkeit des WfMS untersucht werden. Parameter wie z.B. die Verteilungsfunktion für die Bearbeitungszeit simulierter Aktivitäten erlauben die Anpassung des Benchmarks an reale Szenarien.

In der vorgeschlagenen Form liefert der Benchmark in erster Linie Durchsatz- und Durchlaufzeitresultate für ganze Workflows. Um Rückschlüsse auf Schwachstellen in der Implementierung des untersuchten WfMS ziehen zu können, müssen auch Durchsatz und Antwortzeiten von systeminternen Primitiven ermittelt werden. Durch entsprechende Meßpunkte und geeignete Lastparameter können diese Werte während des Ablaufs vollständiger Workflows bestimmt werden.

Der Benchmark definiert keine Strategie für eine verteilte Ausführung. Die Ausführung des Benchmarks kann und soll vielmehr vom WfMS selbst durch geeignete Verteilung optimiert werden. In großen Unternehmen oder "virtuellen Unternehmen" wird es freilich Workflows geben, die dezentralisiert und somit systemtechnisch verteilt ausgeführt werden müssen. Nimmt man z.B. an, daß im Benchmark jedes Lager ein eigenes WfMS betreibt, könnte man sogar die Effizienz von Verteilung und Interoperabilität zwischen WfMS mit dem vorgeschlagenen Benchmark untersuchen.

5.2 Teststufen

In vielen Workflow-Anwendungen werden nur Teile der im Benchmark berücksichtigten Funktionalität genutzt. Um die Leistung von WfMS für solche Anwendungen gezielt zu untersuchen, können verschiedene *Teststufen*, d.h. Abstufungen der Komplexität des Benchmarks, verwendet werden. Denkbar sind z.B. folgende Vereinfachungen:

(1) Weglassen des Zweiges mit interaktiven Aktivitäten, also Beschränkung auf vollautomatische Produktions-Workflows
(2) Weglassen bestimmter Kontrollflußkonstrukte (z.B. Parallelität)
(3) Weglassen von Zeitkonstrukten (z.B. Fristen)

Vereinfachte Benchmarks sind leichter zu realisieren und können auch dann genutzt werden, wenn ein zu untersuchendes System nicht die volle Funktionalität zur Implementierung des gesamten Benchmarks bietet. Umgekehrt könnte der Benchmark auch noch weiter ausgebaut werden zu einer Test- und Meßumgebung für die gesamte Anwendungs-, Infrastruktursoftware und -hardware. In der weitreichendsten Ausbaustufe wären beispielsweise alle externen Applikationen inklusive der zugrundeliegenden Datenstrukturen und Middleware (z.B. Corba) und auch die Nutzung eines Web-Servers zur Kommunikation mit den Clients Gegenstand des Benchmarks.

5.3 Portierbarkeit

Verwendet man ein WfMS, das direkt mit State- und Activitycharts arbeitet, ist die Portierung des Benchmarks sehr einfach. Die Workflow-Spezifikation sollte von einem solchen WfMS direkt interpretierbar oder in Code übersetzbar sein. Lediglich für die simulierten externen Applikationen müssen einfache Treiber geschrieben werden; diese sind jedoch so generisch, daß bei entsprechender Parametrisierung ein Treiber für alle simulierten Applikationen genügt.

Die Portierung des Benchmarks auf andere WfMS ist immer dann einfach, wenn das WfMS ebenfalls einen wohlfundierten "High-Level"-Formalismus für Workflow-Spezifikationen verwendet, beispielsweise Petrinetzvarianten. Die in kommerziellen WfMS verwendeten Spezifikationssprachen sind freilich häufig hybrider Natur, die Elemente verschiedener Formalismen kombinieren – im schlimmsten Fall unter Hinzunahme ausgesprochener Ad-hoc-Konstrukte. Aufgrund unserer Erfahrungen mit verschiedenen Produkten (FlowMark, Staffware, Aris Toolset) stellt sich die Situation allerdings wesentlich besser dar als man pessimistischerweise erwarten könnte. Beispielsweise lassen sich Statechart-Spezifikationen und Ereignis-Prozeß-Ketten, wie sie z.B. dem BPR-Tool Aris zugrundeliegen, weitgehend automatisch ineinander konvertieren [20, 23, 31]. Im Hinblick auf Benchmark-Messungen sind wir dabei, unsere Benchmark-Spezfikation auf das Produkt Staffware zu portieren.

5.4 Notwendiger Meßaufbau

Der Meßaufbau muß eine reproduzierbare Durchführung des Benchmarks gewährleisten. Für jedes zu untersuchende WfMS muß eine eigene Systemumgebung mit allen notwendigen Komponenten erstellt werden. Dies umfaßt auch die Bereitstellung der notwendigen Middleware-Komponenten (z.B. Corba oder ein TP-Monitor), da auch bei simulierten Aktivitäten die entsprechenden Aufrufe zum Starten von Aktivitäten durch das WfMS anfallen. Damit ist die Meßumgebung deutlich komplexer als beispielsweise beim TPC-C-Benchmark.

Im Gegensatz zu typischen Benchmarks für Datenbanksysteme enthält unser Benchmark Aktivitäten von potentiell sehr langer Dauer. Im Benchmark müssen diese Aktivitäten genügend oft durchlaufen werden, um zu Resultaten mit ausreichender Konfidenz zu kommen. Dies führt zu neuen Problemen:

(1) Die Laufzeit des Benchmarks wird sehr lang. Damit ist die Bereitstellung einer dedizierten Meßumgebung problematisch.

(2) Während der Leistungsmessungen sollte sich das WfMS in einem eingeschwungenen Zustand befinden. Dazu muß entweder eine Initialisierungsphase abgewartet werden, bevor die Messungen stattfinden, oder das WfMS muß a priori mit einer definierten Last von aktiven Workflows in unterschiedlichen Zuständen geladen werden. Enthält der Benchmark Aktivitäten von langer Dauer, ist auch die Initialisierungsphase sehr lang. Im Fall der Mahnungen müßte das System mehrere Wochen einschwingen. Das Laden einer definierten Last (ähnlich dem Bulk-Loading eines Datenbanksystems) muß daher möglich sein. Diese Funktionalität wird aber unseres Wissen derzeit von keinem System unterstützt.

6 Ausblick

In dieser Arbeit wurde ein systematischer Benchmark für WfMS vorgeschlagen. Im Hinblick auf die Akzeptanz des Vorschlags sind Messungen an realen System unerläßlich. Solche Messungen sind in Vorbereitung; konkret werden dafür das kommerzielle System Staffware sowie unser eigenes Prototypsystem Mentor [19,31] bzw. Mentor-Lite [30, 32] verwendet. Darüber hinaus wird untersucht, inwieweit man die Workflow-Komponente von SAP R/3 als WfMS in den Benchmarking-Vergleich einbeziehen kann.

7 Literatur

[1] G. Alonso, D. Agrawal, A. Abbadi, C. Mohan: Functionality and Limitations of Current Workflow Management Systems. IEEE Expert 12(5), 1997

[2] T. Bauer, P. Dadam, A Distributed Execution Environment for Large-Scale Workflow Management Systems with Subnets and Server Migration, Proc. Cooperative Information Systems (CoopIS), 1997

[3] A. Bonner, A. Shrufi, S. Rozen, LabFlow-1: a Database Benchmark for High-Throughput Workflow Management, Proc. Extending Database Technology (EDBT), France, 1996

[4] A. Cichoki, A. Helal, M. Rusinkiewicz, D. Woelk: Workflow and Process Automation, Concepts and Technology, Kluwer, 1998

[5] S. Ceri, P. Grefen, G. Sánchez: WIDE - A Distributed Architecture for Workflow Management, Proc. RIDE, 1997
[6] J. Doppelhammer, T. Höppler, A. Kemper, D. Kossmann, Database Performance in the Real World - TPC-D and SAP R/3, Proc. SIGMOD, 1997
[7] A. Geppert, M. Berndtsson, D. Lieuwen, C. Roncancio: Performance Evaluation of Object-Oriented Active Database Management Systems Using the BEAST Benchmark, Theory and Practice of Object Systems (TAPOS) 4:4, October 1998
[8] D. Georgakopoulos, M. Hornick, A. Sheth, An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure, Distributed and Parallel Databases, 3(2), 1995
[9] J Gray (Ed.), The Benchmark Handbook, 2nd Edition, Morgan Kaufmann, 1993
[10] A. Geppert, D. Tombros: Event-based Distributed Workflow Execution with EVE, IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing, The Lake District, Middleware'98, England, September 1998
[11] T. Härder: Informatik Forschung und Entwicklung, Band 12, Heft 2, Springer-Verlag, 1997
[12] D. Harel, State Charts: A Visual Formalism for Complex Systems, Science of Computer Programming Vol.8, 1987
[13] C. Hagen, G. Alonso: Flexible Exception Handling in the OPERA Process Support System, Proc. ICDCS, 1998
[14] D. Harel, E. Gery: Executable Object Modeling with Statecharts, IEEE Computer, 30(7), 1997
[15] R. Jain: The Art of Computer Systems Performance Analysis, John Wiley & Sons, 1991
[16] S. Jablonski, C. Bussler: Workflow-Management, Modeling Concepts, Architecture and Implementation, International Thomson Computer Press, 1996
[17] S. Jablonski, M. Böhm, W. Schulze (Hrsg): Workflow-Management - Entwicklung von Anwendungen und Systemen, dpunkt Verlag, 1997
[18] K. Moore, M. Peterson, A Groupware Benchmark Based on Lotus Notes, Proc. (ICDE), 1996
[19] P. Muth, D. Wodtke, J. Weissenfels, A. Kotz Dittrich, G. Weikum, From Centralized Workflow Specification to Distributed Workflow Execution, Journal of Intelligent Information Systems, Special Issue on Workflow Management, 10(2), 1998
[20] M. Nüttgens, T. Feld, V. Zimmermann: Business Process Modeling with EPC and UML: Transformation or Integration?, in: M. Schader, A. Korthaus (Hrsg.): The Unified Modeling Language - Technical Aspects and Applications, Proc. (Mannheim, Oktober 1997), Workshop des Arbeitskreises "Grundlagen objektorientierter Modellierung" (GROOM) der GI-Fachgruppe 2.1.9 ("Objektorientierte Softwareentwicklung"), Heidelberg 1998
[21] P. O'Neil: Database Performance Measurement, in: A.B. Tucker, (Editor), The Computer Science and Engineering Handbook, CRC Press, 1997
[22] M. Reichert, P. Dadam, $ADEPT_{flex}$ - Supporting Dynamic Changes of Workflows Without Loosing Control, Journal of Intelligent Information Systems, Special Issue on Workflow Management, 10(2), 1998
[23] F. Rump, J. Desel, H. Reichel (Hrsg.): Proc. GI-Workshop "Grundlagen der Parallelität", 1997
[24] A.W. Scheer: Benchmarking Business Process, in: Okino, N.; Tamura, H.; fujii, S. (Hrsg.): Advances in Production Management Systems, Proc. of the 6th IFIP TC5/WG5.7, International Conference on Production Management Systems - APMS, 1996
[25] H. Schuster, P. Heinl: A Workflow Data Distribution Strategy for Scalable Workflow Management Systems. Proc. ACM Symposium on Applied Computing (SAC'97), 1997
[26] Transaction Processing Performance Council, http://www.tpc.org
[27] Unified Modeling Language (UML) Version 1.1,
http://www.rational.com/uml/documentation.html
[28] G. Vossen, J. Becker (Hrsg.): Geschäftprozeßmodellierung und Workflow-Management, Modelle, Methoden, Werkzeuge. International Thomson Publishing, 1996
[29] Workflow Management Coalition. verschiedene Veröffentlichungen,
http://www.aiai.ed.ac.uk/WfMC/
[30] J. Weissenfels, P. Muth, G. Weikum, Flexible Worklist Management in a Light-Weight Workflow Management System, Proc. EDBT Workshop on Workflow Management Systems, 1998
[31] G. Weikum, D. Wodtke, A. Kotz-Dittrich, P. Muth, J. Weissenfels: Spezifikation, Verifikation und verteilte Ausführung von Workflows in MENTOR, Informatik Forschung und Entwicklung, Springer-Verlag, 1997
[32] P. Muth, J. Weissenfels, M. Gillmann, G. Weikum, Integrating Light-Weight Workflow Management Systems within Existing Business Environments, erscheint in Proc. ICDE, 1999